BAB V

IMPLEMENTASI PERANGKAT LUNAK

5.1 Implementasi Secara Umum

Sistem Optimasi Tata letak barang di gudang dengan Algoritma Genetika ini diimplementasikan dengan bahasa pemrograman visual, yaitu Borland Delphi 6.0. Tahap implementasi merupakan akumulasi dari seluruh analisa sistem yang dilakukan diawal. Salah satu tahap di mana sistem dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat benar-benar sesuai dengan yang diharapkan. Hal-hal yang berkaitan dengan fungsi dan prosedure juga kebutuhan akan pengembangan sistem dimanifestasikan kedalam sistem nyata.

Sebelum program diimplementasikan, maka program harus bebas dari kesalahan. Kesalahan program yang mungkin terjadi antara lain karena kesalahan penulisan (*coding*), kesalahan proses, atau kesalahan logika.

Sistem Optimasi ini dapat berjalan dengan baik, apabila memenuhi standar minimal dari perangkat keras (*hardware*). Perangkat keras yang digunakan minimal memiliki spesifikasi sebagai berikut:

1. Satu unit komputer dengan spesifikasi minimum Prosesor celeron 366 MHz, RAM 128 MB, Hard Disk 2 GB.
2. Satu unit printer Dot Matrix, LaserJet atau Deskjet (optional) dan kertas
3. Monitor VGA atau SVGA
4. Mouse

5. Keyboard

Perangkat lunak yang dibutuhkan untuk pengembangan dan implementasi sistem optimasi tata letak barang digudang dengan algoritma genetika ini menggunakan:

1. Sistem Operasi Windows 9.x atau 2000 atau yang lebih tinggi
2. Bahasa Pemrograman Borland Delphi 6.0 untuk pembuatan perangkat lunak
3. Microsoft Word 97 atau yang lebih tinggi.
4. Software Notepad

5.2 Alasan Pemilihan Bahasa Pemrograman

Beberapa alasan penggunaan bahasa pemrograman visual Borland Delphi 6.0 antara lain adalah :

1. Borland Delphi 6.0 adalah bahasa pemrograman yang dirancang untuk bekerja di dalam *platform* Microsoft Windows.
2. Borland Delphi menggunakan bahasa *Object* Pascal yang berarti merupakan bahasa pemrograman terstruktur, sehingga memudahkan dalam *coding* maupun *debugging*.
3. Borland Delphi juga mempunyai kemampuan operasi numerik serta menyediakan fasilitas grafik dan Windows API yang sangat menunjang dalam pembuatan perangkat lunak.
4. Tersedianya banyak komponen-komponen visual yang cukup representatif untuk digunakan dalam membangun aplikasi berbasis *windows*. Dan

5.4 Tahap Pembuatan Perangkat Lunak

Dalam pembuatan perangkat lunak sistem Optimasi Tata Letak Barang Digudang Dengan Algoritma Genetika dengan bahasa pemrograman Borland Delphi 6.0 dibangun dengan beberapa tahapan, yang dapat dibedakan menjadi 3 tahap yaitu :

1. Tahapan pembuatan basis data

   Tahapan ini dilakukan untuk menyusun tabel-tabel yang digunakan untuk menyimpan data yang diperlukan dari hasil analisa sebelumnya.

2. Tahap Pemrograman Visual

   Pada tahap ini yang dilakukan adalah merancang *form* yang akan digunakan program serta kontrol kontrol yang diperlukan. Perancangan tersebut ditangani oleh paket-paket yang disediakan oleh Borland Delphi 6.0.

3. Tahap Penulisan Kode (*Coding*)

   Pada tahap ini yang dilakukan adalah menuliskan kode – kode yang diimplemetasikan dengan *procedure* untuk selanjutnya ditempatkan atau dipanggil pada kontrol-kontrol (*object*) yang dipakai. Penulisan kode ini dilakukan dengan menggunakan *Text Editor* milik Borland Delphi 6.0.

4. Tahap *Debugging*

   Pada tahap ini dilakukan pengujian terhadap perangkat lunak yang dibuat. Metode pengujian yang digunakan ialah metode *White Box*, dimana tiap langkah yang menghasilkan *output* ditulis ulang ke dalam *Log File* dan

2. Tombol *Genetika*

   Tombol ini digunakan untuk menampilkan / mengaktifkan *form* proses algoritma genetika.

3. Tombol *Visual 2D*

   Tombol ini digunakan untuk menampilkan / mengaktifkan *form* visual.

4. Tombol *Laporan*

   Tombol ini digunakan untuk menampilkan / mengaktifkan *form* Laporan yang merupakan solusi sebagai hasil akhir dari proses algoritma genetika.

5. Tombol *Kembali ke awal*

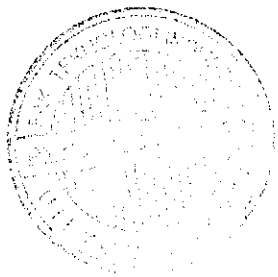
   Tombol ini untuk menampilkan form menu seperti semula, ketika form pertama kali diaktifkan.

5 Tombol *Keluar*

   Tombol ini digunakan untuk keluar dari program.

5.5.2 Form Data

Form *Data* digunakan untuk menginisialisasi data yang akan diproses. Disini terdiri dari bagian inisialisasi *File*, input yang terdiri dari dimensi barang dan gudang



Gambar 5.2 Tampilan keseluruhan Form *Data*

Pada bagian Form *Data* terdapat beberapa tombol yaitu :

1. Tombol *Baru*

   Tombol ini digunakan untuk membuat file baru. Jika tombol ini ditekan maka akan ditampilkan sebuah *form* yaitu *form* IsiData.

2. Tombol *Buka*

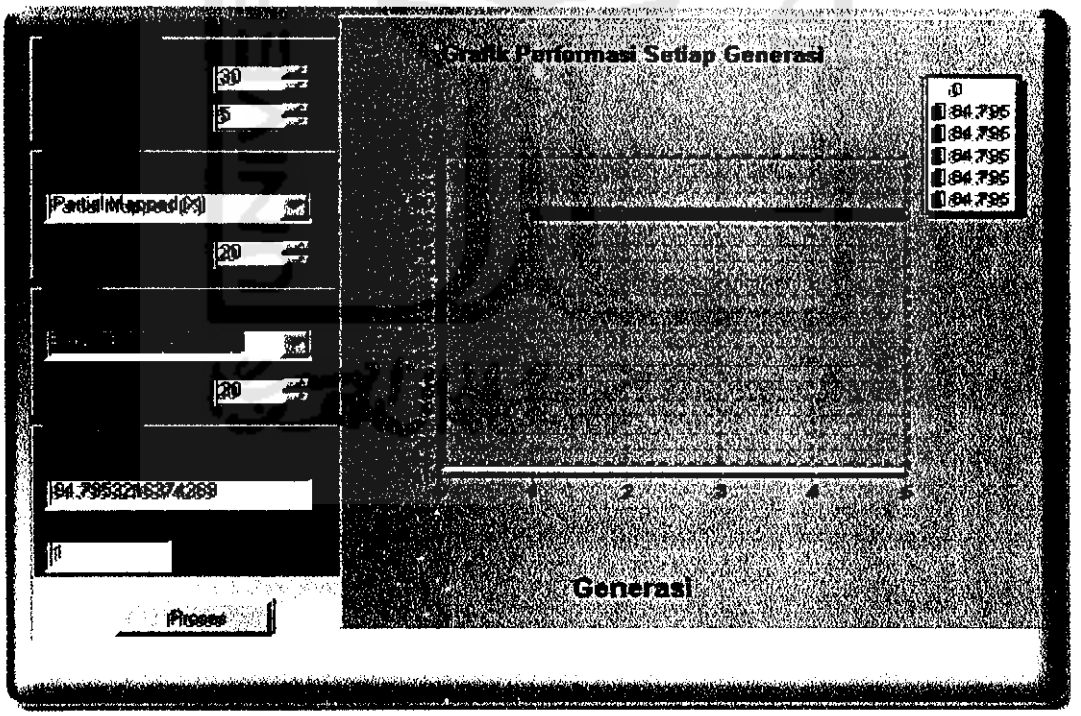
   Tombol ini digunakan untuk membuka *file* yang telah disimpan atau dibuat sebelumnya. Jika tombol ini ditekan maka akan ditampilkan *Open Dialog*, kemudian dipilih nama *file* yang akan dibuka.

3. Tombol *Simpan*

Tombol ini digunakan untuk menyimpan *file* atau data yang telah diinputkan ke dalam sistem. Jika tombol ini ditekan, maka akan ditampilkan *Save Dialog*, kemudian diisikan nama *file* baru.

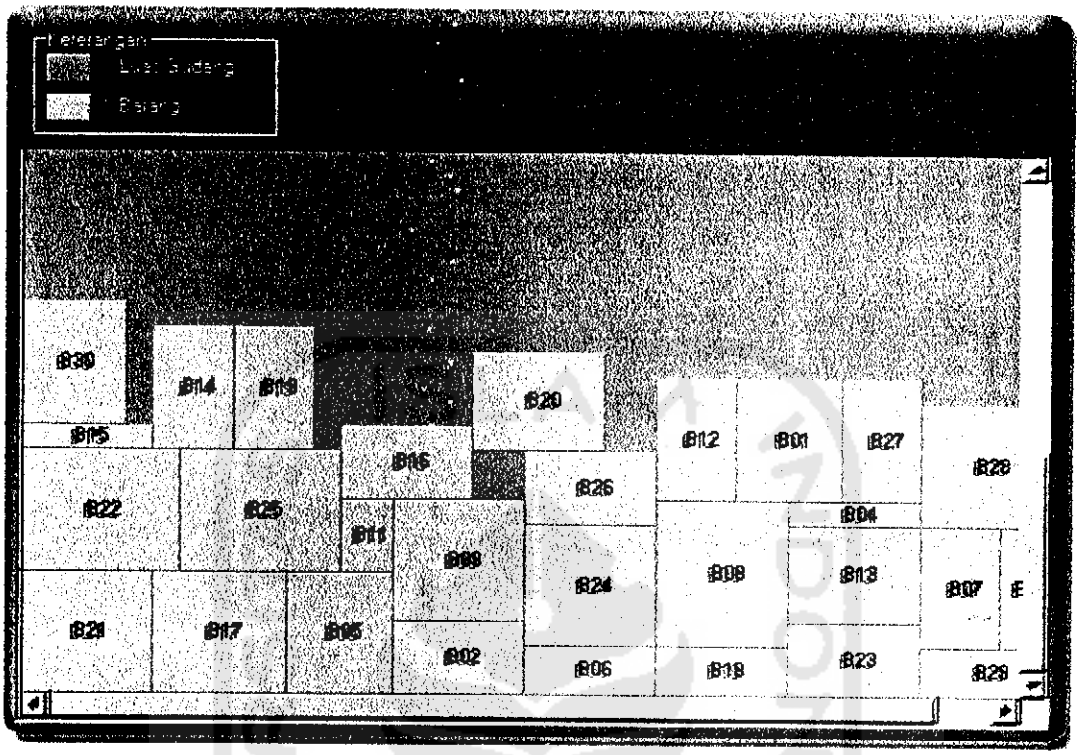
5.5.3 Form *Genetika*

Form *Genetika* ini digunakan untuk memproses data yang telah diinisialisasi pada form *Data*. Pada bagian ini dibagi menjadi 5 bagian yaitu bagian parameter algoritma genetika, Metode *CrossOver*, Metode Mutasi, *Fitness* terbaik (Solusi), dan grafik nilai *fitness* terbaik untuk tiap generasi.



Gambar 5.3 Tampilan keseluruhan Form *Genetika*

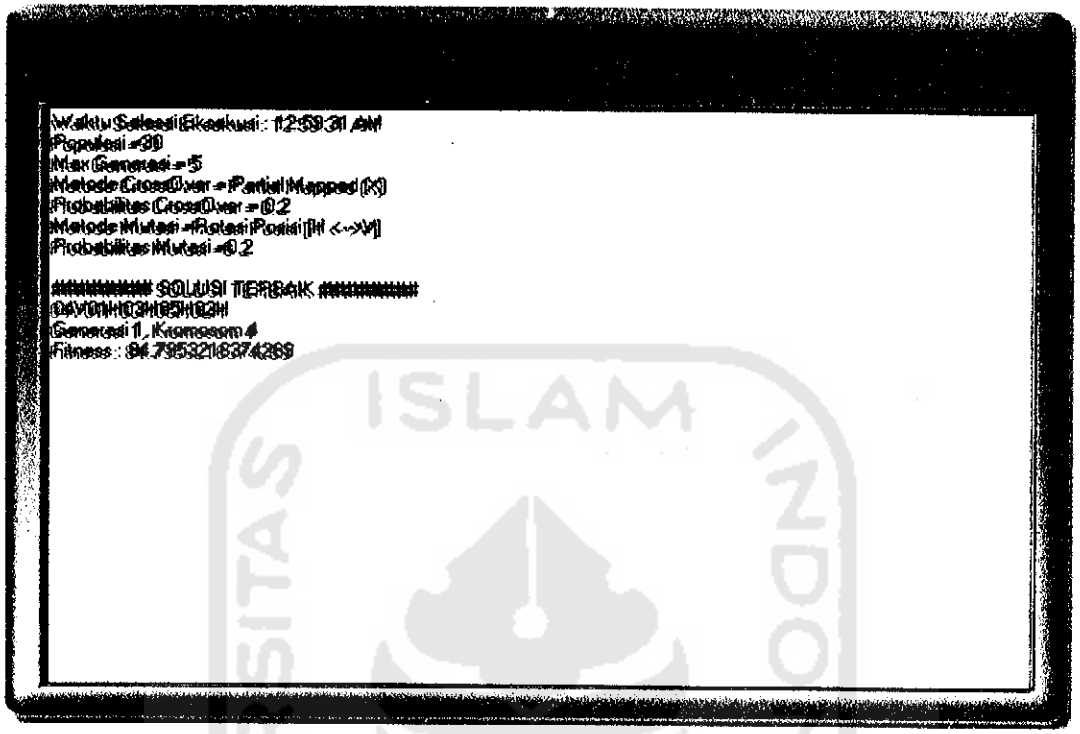
Pada bagian parameter algoritma genetika, terdapat beberapa jenis masukan yaitu Ukuran Populasi, Jumlah Generasi, Probabilitas *CrossOver* beserta



Gambar 5.4 Tampilan Form *Visual*

5.5.5 Form *Laporan*

Pada form laporan akan ditampilan hasil akhir dari proses algoritma genetika. Dalam laporan akan ditampilkan durasi waktu yang dibutuhkan dari proses, beberapa parameter algoritma genetika, metode *crossover* dan mutasi yang digunakan dalam proses pencarian. Solusi nilai *fitness* terbaik beserta urutan tata letak barang.



Gambar 5.5 Tampilan *Laporan* Proses *Algoritma Genetika*

5.6 Prosedur dan Algoritma

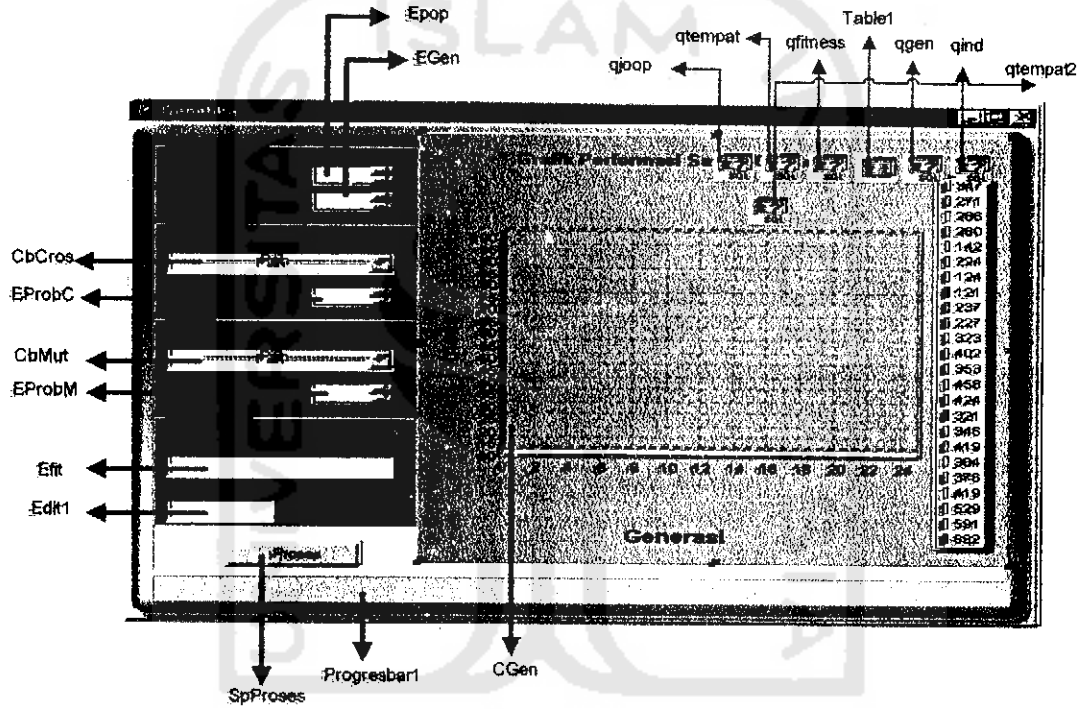
5.6.1 Teknik Pemrograman

Teknik yang digunakan adalah teknik prosedural, artinya sistem secara keseluruhan menggunakan sub sistem aturan yang saling berkaitan dengan lainnya. Alat pemrograman yang digunakan adalah Borland Delphi 6.0 yang cukup mudah digunakan dengan komponen-komponen pendukungnya, serta mendukung teknik pemrograman prosedural.

Dengan menggunakan Borland Delphi 6.0 secara otomatis file yang dibuat akan terbentuk menjadi empat jenis file, yaitu file dengan ektensi *dpr,pas,dfm* dan *dcu*. Jika dikompilasi maka akan menghasilkan file dengan ektensi *exe*.

5.6.2 Prosedur – Prosedur dalam program

Pemrograman di dalam perangkat lunak Optimasi tata letak barang digudang dengan algoritma genetika ini terdiri atas prosedur-prosedur yang dapat memudahkan dalam pembuatan program. Berikut ini beberapa sebagian prosedur – prosedur yang digunakan beserta pembahasannya.



Gambar 5.6 Objek form Genetika

5.6.2.1 Prosedur Pada Form Genetika

Prosedur Proses ini digunakan untuk melakukan optimasi atau pencarian solusi tata letak barang digudang berdasarkan data masukan pada form *Data*. Di dalam prosedur Proses ini terdapat 2 bagian utama, yaitu pendeklarasian variabel, iterasi. Penjelasannya sebagai berikut:

Fungsi untuk menentukan populasi awal atau generasi awal.

```
function TGenetika.ind(gen,pop:integer): string;

var i,noangka,pos:integer;
    temp,tempos,individu:string;
begin
qind.SQL.Clear;
qind.SQL.Add('select * from tbarang');
qind.Active:=True;
if not qind.Eof then
random(qind.RecordCount);
i:=1;

while not qind.Eof do
begin
  //menentukan angka
  noangka:=random(qind.RecordCount);
  if noangka =0 then
  noangka:=noangka+1;
  if noangka < 10 then
    temp:='0'+inttostr(noangka)
  else
    temp:=inttostr(noangka);
  {pos := random(1);
  pos :=random(1);

  if i <> 1 then
  temp:=cariangka(individu,temp,qind.RecordCount+1);
  pos :=random(2);
  //menetukan posisi barang
  if pos = 0 then
    tempos := 'H'
  else
    tempos:='V';

  qjoop.SQL.Clear;
  qjoop.SQL.Add('update tbarang set pos = ' + quotedstr(tempos) +
' where kode_barang=' + quotedstr(temp));
  qjoop.ExecSQL;
  individu:=individu+temp+tempos;
  i:=i+1;
  qind.Next;
  end;
ind :=individu;
end;
```

Pada fungsi diatas untuk menentukan populasi awal atau sebagai generasi pertama, dimana penentuan populasi awal ini dilakukan secara random. Sebagai kromosom adalah kode barang beserta posisinya yang dipilih secara random juga.

Pada fungsi ini juga memanggil fungsi lainnya yang digunakan untuk menentukan penomoran kromosom.

fungsi untuk menghitung nilai fitness

```
function TGenetika.fitness(strind:string):real;
var i:integer;
    P,L,pgudang,lgudang,pjg,lbr,p1,p2,l1,l2:integer;
    kode,kodekanan : string;
    ls,lb,lg,lga,ordx,ordy,xkanan,ykanan:integer;
    hbs1 : boolean;
begin
lbarang:= 0;
hbs1 :=false;
ls:=0;
lb:=0;
i:=1;
//set semua brg belum teralokasi
QGen.SQL.Clear;
QGen.SQL.Add('update tbarang set packed = ' + quotedstr('T') +
',ordinaty=0,ordinatx=0' );
QGen.ExecSQL;

QGen.SQL.Clear;
QGen.SQL.Add('Select * from tgudang');
QGen.Active:=true;
pgudang := qgen.fieldbyname('Panjang').AsInteger;
lgudang:=Qgen.fieldbyname('Lebar').AsInteger;
lg:=(pgudang*lgudang);
l:= Qgen.fieldbyname('Lebar').AsInteger;
lga:=lg;
QGen.SQL.Clear;
  while (i <= length(strind)) do
   begin
     kode:= strind[i] + strind[i+1];
     qgen.Active:=false;
     qgen.SQL.Clear;
     QGen.SQL.Add('Select * from tbarang where kode_barang=' +
     quotedstr(kode));
     QGen.Active:=true;
     lb:=qgen.fieldbyname('Panjang').AsInteger                    *
     qgen.fieldbyname('Lebar').AsInteger ;
     pjg:= qgen.fieldbyname('Panjang').AsInteger;
     lbr:=qgen.fieldbyname('lebar').AsInteger;
     if hbs1 = false then
     begin
        if lg > lb then
                lg:=lg-lb;
        if i=1 then //utk baris pertama
        begin
          ordx:=0;
          ordy:=0;
        end;
```

```
        if (((ordx + QGen.fieldbyname('Panjang').AsInteger) < l)
        and       (strind[i+2]='V'))       or       (((ordx       +
        QGen.fieldbyname('Lebar').AsInteger)     <     1)     and
        (strind[i+2]='H')) then
       begin
        qfitness.SQL.Clear;
        qfitness.SQL.Add('update            tbarang            set
        Ordinatx='+inttostr(ordx)+',ordinaty='+inttostr(ordy)+',p
        acked    =    '    +    quotedstr('Y')    +    '    where
        Kode_Barang='+Quotedstr(kode));
        QFitness.ExecSQL;
       end;
       if strind[i+2]='V' then
       begin
              ordx:=ordx + QGen.fieldbyname('Panjang').AsInteger;
       end
       else
       begin
              Ordx:=ordx + QGen.fieldbyname('Lebar').AsInteger;
       end;
       if ordx >= l then
       begin
       hbsl:=true;
        if strind[i+2]='V' then
        begin
              lbr := QGen.fieldbyname('Panjang').AsInteger;
              pjg:= QGen.fieldbyname('Lebar').AsInteger;
        end
        else
        begin
              lbr:= QGen.fieldbyname('Lebar').AsInteger;
              pjg:= QGen.fieldbyname('Panjang').AsInteger;
        end;
           caritempat(kode,pjg,lbr,pgudang,lgudang);
      end;
    end
    else //baris selanjutnya
    begin;
       if strind[i+2]='V' then
       begin
              lbr := QGen.fieldbyname('Panjang').AsInteger;
              pjg:= QGen.fieldbyname('Lebar').AsInteger;
       end
       else
       begin
              lbr:= QGen.fieldbyname('Lebar').AsInteger;
              pjg:= QGen.fieldbyname('Panjang').AsInteger;
       end;
       caritempat(kode,pjg,lbr,pgudang,lgudang);
    end;
 i:=i+3;
 ls:=ls+lb;
 qgen.Next;
end;
QGen.SQL.Clear;
```

```
QGen.SQL.Add('Select * from tbarang where packed = ' +
quotedstr('Y'));
qgen.Active:=true;
while not qgen.Eof do
 begin
  lbarang := lbarang + (qgen.fieldbyname('panjang').AsInteger *
  qgen.fieldbyname('lebar').AsInteger) ;
  qgen.Next;
end;
QGen.SQL.Clear;
QGen.SQL.Add('Select ordinatx + panjang as lgudang from tbarang
where packed = ' + quotedstr('Y') + ' and pos = ' +
quotedstr('H') + ' order by lgudang desc');
qgen.Active:=true;
l1 := qgen.fieldbyname('lgudang').AsInteger ;
QGen.SQL.Clear;
QGen.SQL.Add('Select ordinatx + lebar as lgudang from tbarang
where packed = ' + quotedstr('Y') + ' and pos = ' +
quotedstr('V') + ' order by lgudang desc');
 qgen.Active:=true;
 l2 := qgen.fieldbyname('lgudang').AsInteger ;
 if l1 > l2 then
   lgudang := l1
 else
    lgudang := l2;
QGen.SQL.Clear;
QGen.SQL.Add('Select ordinaty + panjang as pgudang from tbarang
where packed = ' + quotedstr('Y') + ' and pos = ' +
quotedstr('V') + ' order by pgudang desc');
 qgen.Active:=true;
p1 := qgen.fieldbyname('pgudang').AsInteger ;
QGen.SQL.Clear;
QGen.SQL.Add('Select ordinaty + lebar as pgudang from tbarang
where packed = ' + quotedstr('Y') + ' and pos = ' +
quotedstr('H') + ' order by pgudang desc');
qgen.Active:=true;
 p2 := qgen.fieldbyname('pgudang').AsInteger ;
 if p1 > p2 then
    pgudang := p1
 else
    pgudang := p2;

    lga := lgudang * pgudang;
    fitness:=lbarang/lga;
end;
```

Fungsi diatas digunakan untuk menghitung nilai fitnes dengan menghitung jumlah luas barang dibagi dengan luas okupansinya.

Bagian perhitungan probabilitas relatif dan komulatif

```
procedure tgenetika.hitung_eval(var gen : integer);
var i : integer;
    totfit,rel,kum : real;
begin
qgen.SQL.Clear;
qgen.SQL.Add('select sum(fitness) from tgen where gen = ' +
inttostr(gen));
qgen.Active:=true;
totfit := qgen.Fields[0].AsFloat;
Laporan.RE.Lines.Add('Total Fitness = ' + qgen.Fields[0].AsString
);
Laporan.RE.Lines.Add('');
qgen.SQL.Clear;
qgen.SQL.Add('select strind,pop,gen,fitness from tgen where gen =
' + inttostr(gen) +  ' order by pop');
qgen.Active:=true;
Laporan.RE.Lines.Add('== Probabilitas Relatif & Kumulatif ==');
kum := 0;
i:=1;
while not qgen.Eof do
begin
  rel := qgen.fieldbyname('fitness').AsFloat / totfit;
  kum := kum + rel;
  Laporan.RE.Lines.Add('[' + inttostr(i) +']  Prob Relatif => ' +
floattostr(rel) + ' -- Prob Kumulatif => ' + floattostr(kum));
  qjoop.SQL.Clear;
  qjoop.SQL.Add('update tgen set probr=' + floattostr(rel) +
',probk='+ floattostr(kum) + ' where gen=' + inttostr(gen) + ' and
pop=' + inttostr(i));
  qjoop.ExecSQL;
  qgen.Next;
  i:=i+1;
end;
end;
```

prosedure diatas untuk menghitung fitnes total fitness, probabilitas relatif dan prob kumulatif

Procedure *Order Crossover* (OX) pada form *Genetika*

```
procedure tgenetika.silang_ox(induk1,induk2 : string);
var
  i, j, l, k, x : longint;
  relasi_map : array of array of string;
  nobrg1,nobrg2 : integer;
  substring_acak1, substring_acak2,test1,test2,joop,   joop1,joop2
: string;

  temp2 : char;
```

```
  temp : integer;
begin
  i := 0;
      {pengacakan substring}
      repeat
        random(jmlbrg);
        repeat
        nobrg1:=random(jmlbrg);
        until nobrg1<>0;
        repeat
        nobrg2:=random(jmlbrg);
        until nobrg2<>0;
        if nobrg1 > nobrg2 then
        begin
          temp := nobrg1;
          nobrg1 := nobrg2;
          nobrg2 := temp;
        end;
      until (nobrg1 <> nobrg2) and ((nobrg2 - nobrg1) > 1);
      {pengsisian child1 dengan substring dari parent 1 dan child2
      dengan substring dari parent2}
      joop1:='';
      joop2:='';
      SetLength(joop1,((jmlbrg * 3)) );
      SetLength(joop2,((jmlbrg * 3)));
      for i := 1 to (nobrg1*3) do
      begin
        joop1[i]:= 'x';
        joop2[i]:= 'x';
      end;
      for i := (nobrg1*3) + 1 to (jmlbrg*3) do
      begin
        joop1[i]:= 'x';
        joop2[i]:= 'x';
      end;
      for j := (((nobrg1) * 3) + 1) to (((nobrg2) * 3)) do
      begin
        joop1[j] := induk1[j];
        joop2[j] := induk2[j];
      end;

      {induk yang telah mengalami penyilangan untuk posisi 1}
      j:=1;
      x:=1;
      while j <=  (jmlbrg * 3 )  do
      begin
        k := ((nobrg1) * 3) + 1;
        joop:='';
        while k <=((nobrg2) * 3) do
        begin
           if(induk2[j]+induk2[j+1])<>(joop1[k] + joop1[k+1]) then
             begin
              k:= k + 3;
              joop := (induk2[j] + induk2[j+1] + induk2[j+2]);
             end
             else
```

```
            begin
             k:= ((nobrg2) * 3) + 1;
             joop:='';
            end;
        end;

        if (joop <> '')  then
        begin
          joop1[x] := joop[1];
          joop1[x+1] := joop[2];
          joop1[x+2] := joop[3];
          x:=x+3;
        if(x>=((nobrg1 * 3)+1)) and (x < ((nobrg2 * 3) + 1))  then
           x:= (nobrg2 * 3) + 1;
          if x> (jmlbrg * 3 ) then
            j:=(jmlbrg * 3 );
        end;
        j:=j+3;
      end;
      {induk yang telah mengalami penyilangan untuk posisi 2}
      j:=1;
      x:=1;
      while j <=  (jmlbrg * 3 )  do
      begin
        k := ((nobrg1) * 3) + 1;
        joop:='';
        while k <=((nobrg2) * 3) do
        begin
           if(induk1[j]+induk1[j+1])<>(joop2[k] + joop2[k+1]) then
             begin
              k:= k + 3;
              joop := (induk1[j] + induk1[j+1] +   induk1[j+2]);
             end
             else
             begin
              k:= ((nobrg2) * 3) + 1;
              joop:='';
             end;
        end;
        if (joop <> '')  then
        begin
          joop2[x] := joop[1];
          joop2[x+1] := joop[2];
          joop2[x+2] := joop[3];
          x:=x+3;
          if(x >=((nobrg1)*3+1))and (x < ((nobrg2 * 3) + 1))  then
           x:= (nobrg2 * 3) + 1;
          if x> (jmlbrg * 3 ) then
           j:=(jmlbrg * 3 );
        end;
        j:=j+3;
      end;
      child1 := joop1;
      child2 := joop2;
end;
```

Prosedur diatas digunakan untuk menjalankan metode *Order CrossOver*. Pada metode ini memilih substring dari parent yang akan disilangkan kemudian mengcopy ke *offspring* yang akan digenerasi dengan posisi yang sama. Selanjutnya abaikan gen dengan nilai yang sama dengan yang sudah pada offspring, tempatkan sisa substring parent lainnya dari kiri kekanan menurut aturan dalam menggenerasi *offspring*.

Procedure *Partial Mapped CrossOver* **(PMX) pada form** *Genetika*

```
Procedure tgenetika.silang_pmx(induk1,induk2 : string);
var
  i, j, l, k : integer;
  relasi_map : array of array of string;
  nobrg1,nobrg2 : integer;
  substring_acak1,substring_acak2,test1,test2,joop,joop1,joop2:
string;
  temp2 : char;
  temp : integer;
begin
        repeat
        random(jmlbrg);
        repeat
        nobrg1:=random(jmlbrg);
        until nobrg1<>0;
        repeat
        nobrg2:=random(jmlbrg);
        until nobrg2<>0;
        if nobrg1 > nobrg2 then
        begin
          temp := nobrg1;
          nobrg1 := nobrg2;
          nobrg2 := temp;
        end;
      until nobrg1 <> nobrg2;
      setlength(relasi_map,0,0);
      setlength(relasi_map,jmlbrg*3,jmlbrg*3);
        test1 := '';
        test2:= '';
      l := 1;
      for j := nobrg1  to nobrg2  do
      begin
        test1 := test1 + induk1[l] + induk1[l+1] + induk1[l+2];
        test2 := test2 + induk2[l] + induk2[l+1] + induk2[l+2];
        l := l + 3;

      end;
```

```
        l := 1;
        i:=1;
        for j := nobrg1  to nobrg2  do
        begin
          relasi_map[i-1,0] := induk1[l] + induk1[l+1];
          temp2:=induk2[l];
          induk2[l]:=induk1[l];
          induk1[l]:=temp2;
          relasi_map[i-1,1] := induk2[l] + induk2[l+1];
          temp2:=induk2[l+1];
          induk2[l+1]:=induk1[l+1];
          induk1[l+1]:=temp2;
          l := l + 3;
          i:=i+1;

        end;
        j:=1;

        joop1 := test1;
        joop2 := test2;
        for i:=(nobrg2 - nobrg1 + 2) to (jmlbrg - 1) do
        begin
           repeat
           k:= random(jmlbrg);
           until k <> 0;

           if k < 10 then
             joop:='0'+inttostr(k)
           else
           joop:=inttostr(k);
           joop1:=joop1+cariangka(joop1,joop,jmlbrg) + induk1[i*3];
           joop2:=joop2+cariangka(joop2,joop,jmlbrg) + induk2[i*3];

        end;
           joop1:=joop1+cariangka(joop1,'02',jmlbrg+1)+induk1[i*3];
           joop2:=joop2+cariangka(joop2,'02',jmlbrg+1)+induk2[i*3];
      child1 := joop2;
      child2 := joop1;

end;
```

Pada procedure diatas untuk menjalankan metode *Partial Mapped Crossover* (PMX) pada proses genetika. PMX menggunakan prosedur khusus untuk mengatasi terjadinya *offspring* yang ilegal dari persilangan dua titik. Inti dari PMX adalah persilangan dua titik ditambah dengan prosedur perbaikan

Procedure *Position Based CrossOver* (PBX) **pada form** *Genetika*

```
procedure tgenetika.silang_pbx(induk1,induk2 : string);
var
  i, j, l, k,m, x : longint;
  nobrg1,nobrg2 : integer;
  substring_acak1, substring_acak2,joop1,joop2 : string;
  temp,jml_acak,acak : integer;
  temp1,temp2 : array of string;
  found,found2 : boolean;
  joop : array of integer;
begin
      {pengacakan jumlah node acak}
      repeat
        jml_acak := random(jmlbrg );
      until (jml_acak <> 0) and (jml_acak < (jmlbrg - 2));

      {pengsisian child1 dengan substring dari parent 1 dan child2
      dengan substring dari parent2}
      setlength(joop1,1);
      setlength(joop1,jmlbrg * 3 );
      setlength(joop2,1 );
      setlength(joop2,jmlbrg * 3);
      setlength(joop,1);
      setlength(joop,jml_acak +1);
      setlength(temp1,1);
      setlength(temp1,jml_acak + 1);
      setlength(temp2,1);
      setlength(temp2,jml_acak + 1);
      {pengacakan node acak}
      for j := 1 to jml_acak  do
      begin
        repeat
          repeat
          acak := random(jmlbrg);
          until acak <> 0;
          found := false;
          l := 1;
          while ((l <= jml_acak) and (not found)) do
          begin
            if joop[l] = acak then
              found := true
            else
              l := l + 1;
          end;
        until found = false;
        joop[j] := acak;
      end;
      for i := 1 to (jmlbrg*3) do
      begin
        joop1[i]:= 'x';
        joop2[i]:= 'x';
      end;

      for j := 1 to jml_acak  do
      begin
```

```
        joop1[((joop[j]-1) * 3) + 1 ]:= induk2[((joop[j]-1)*3)+1];
        joop1[((joop[j]-1) * 3) + 2 ]:= induk2[((joop[j]-1)*3)+2];
        joop1[((joop[j]-1) * 3) + 3 ]:= induk2[((joop[j]-1)*3)+3];
        temp1[j]:=joop1[((joop[j]-1)*3)+1]+joop1[((joop[j]-)*3)+2];
        joop2[((joop[j]-1)*3)+1]:=induk1[((joop[j]-1)*3)+1];
        joop2[((joop[j]-1)*3)+2]:=induk1[((joop[j]-1)*3)+2];
        joop2[((joop[j]-1)*3)+3]:=induk1[((joop[j]-1)*3)+3];
        temp2[j]:=joop2[((joop[j]-1)*3)+1]+joop2[((joop[j]-1)*3)+2
];
      end;
     {induk yang telah mengalami penyilangan untuk posisi 1}
      j:=1;
      //perulangan pada string tempchild
      while j <= ((jmlbrg) * 3) do
      begin
        m:=1;
        //perulangan pada string induk
        while m <= ((jmlbrg) * 3) do
        begin
          found := false;
          l:=1;
          while l <= ((jmlbrg) * 3) do
          begin
            //cek apakah substring induk ada pada tempchild
            if induk1[m]+induk1[m+1] = joop1[l] + joop1[l+1] then
            begin
              found := true;
              l:=jmlbrg* 3;
            end;
            l:=l+3;
          end;
          if not found then
          begin
             if(joop1[j]+joop1[j + 1] + joop1[j + 2]) = 'xxx' then
                begin
                   joop1[j] := induk1[m];
                   joop1[j+1] := induk1[m+1];
                   joop1[j+2] := induk1[m+2];
                end;
            m:=jmlbrg*3;
          end;
          m:=m+3;
        end;
        j:=j+3;
      end;

      {induk yang telah mengalami penyilangan untuk posisi 2}
      j:=1;
      //perulangan pada string tempchild
      while j <= ((jmlbrg) * 3) do
      begin
        m:=1;
        //perulangan pada string induk
        while m <= ((jmlbrg) * 3) do
        begin
```

```
            found := false;
            l:=1;
            while l <= ((jmlbrg) * 3) do
            begin
              //cek apakah substring induk ada pada tempchild
              if induk2[m]+ induk2[m+1] = joop2[l] + joop2[l+1] then
              begin
                found := true;
                l:=jmlbrg* 3;
              end;
              l:=l+3;
            end;
            if not found then
            begin
                  if(joop2[j]+joop2[j+1]+joop2[j+2])='xxx' then
                  begin
                  joop2[j] := induk2[m];
                  joop2[j+1] := induk2[m+1];
                  joop2[j+2] := induk2[m+2];
                  end;
              m:=jmlbrg*3;
            end;
            m:=m+3;
          end;
          j:=j+3;
        end;
      childl:=joop1;
      child2:=joop2;

end;
```

Pada prosedur diatas Pada intinya sejenis representasi *crossover* pada umumnya, dengan prosedur pembenahan. *Position Based* disebut juga variasi dari OX dengan pemilihan *substring* secara tidak urut

Procedure *Order Based CrossOver* (OBX) **pada form** *Genetika*

```
procedure tgenetika.silang_obx(induk1,induk2 : string);
var
  i, j, l, k,m, x : longint;
  nobrg1,nobrg2 : integer;
  substring_acak1, substring_acak2,joop1,joop2 : string;
  temp,jml_acak,acak : integer;
  temp1,temp2 : array of string;
  found,found2 : boolean;
  joop : array of integer;
begin
      {pengacakan jumlah node acak}
      repeat
        jml_acak := random(jmlbrg );
      until (jml_acak <> 0) and (jml_acak < (jmlbrg - 2));
```

```
{pengsisian child1 dengan substring dari parent 1 dan child2
dengan substring dari parent2}
setlength(joop1,1);
setlength(joop1,jmlbrg * 3 );
setlength(joop2,1 );
setlength(joop2,jmlbrg * 3);
setlength(joop,1);
setlength(joop,jml_acak +1);
setlength(temp1,1);
setlength(temp1,jml_acak + 1);
setlength(temp2,1);
setlength(temp2,jml_acak + 1);
{pengacakan node acak}
for j := 1 to jml_acak  do
begin
  repeat
    repeat
    acak := random(jmlbrg);
    until acak <> 0;
    found := false;
    l := 1;
    while ((l <= jml_acak) and (not found)) do
    begin
      if joop[l] = acak then
        found := true
      else
        l := l + 1;
    end;
  until found = false;
  joop[j] := acak;
end;
for i := 1 to (jmlbrg*3) do
begin
  joop1[i]:= 'x';
  joop2[i]:= 'x';
end;

for j := 1 to jml_acak  do
begin
  joop1[((joop[j]-1)*3)+1]:= induk1[((joop[j]-1) * 3) + 1 ];
  joop1[((joop[j]-1)*3)+2]:= induk1[((joop[j]-1) * 3) + 2];
  joop1[((joop[j]-1)*3)+3]:= induk1[((joop[j]-1) * 3) + 3];
  temp1[j]:=joop1[((joop[j]-1)*3)+1]+joop1[((joop[j]-
1)*3)+2];
  joop2[((joop[j]-1)*3)+1]:=induk2[((joop[j]-1)*3)+1 ];
  joop2[((joop[j]-1)*3)+2]:=induk2[((joop[j]-1)*3)+2];
  joop2[((joop[j]-1)*3)+3]:=induk2[((joop[j]-1)*3)+3];
  temp2[j]:=joop2[((joop[j]-1)*3)+1]+joop2[((joop[j]-
1)*3)+2];
end;

{induk yang telah mengalami penyilangan untuk posisi 1}
j:=1;
//perulangan pada string tempchild
while j <= ((jmlbrg) * 3) do
```

```
begin
  m:=1;
  //perulangan pada string induk
  while m <= ((jmlbrg) * 3) do
  begin
    found := false;
    l:=1;
    while l <= ((jmlbrg) * 3) do
    begin
      //cek apakah substring induk ada pada tempchild
      if induk2[m]+induk2[m+1] = joop1[l] + joop1[l+1] then
      begin
        found := true;
        l:=jmlbrg* 3;
      end;
      l:=l+3;
    end;
    if not found then
    begin
         if(joop1[j]+joop1[j+1] + joop1[j + 2]) = 'xxx' then
          begin
             joop1[j] := induk2[m];
             joop1[j+1] := induk2[m+1];
             joop1[j+2] := induk2[m+2];
          end;
      m:=jmlbrg*3;
    end;
    m:=m+3;
  end;
  j:=j+3;
end;

{induk yang telah mengalami penyilangan untuk posisi 2}
j:=1;
//perulangan pada string tempchild
while j <= ((jmlbrg) * 3) do
begin
  m:=1;
  //perulangan pada string induk
  while m <= ((jmlbrg) * 3) do
  begin
    found := false;
    l:=1;
    while l <= ((jmlbrg) * 3) do
    begin
      //cek apakah substring induk ada pada tempchild
      if induk1[m]+induk1[m+1] = joop2[l] + joop2[l+1] then
      begin
        found := true;
        l:=jmlbrg* 3;
      end;
      l:=l+3;
    end;
    if not found then
    begin
          if(joop2[j]+joop2[j+1]+joop2[j+2]) = 'xxx' then
```

```
                begin
                joop2[j] := induk1[m];
                joop2[j+1] := induk1[m+1];
                joop2[j+2] := induk1[m+2];
                end;
              m:=jmlbrg*3;
            end;
            m:=m+3;
          end;
          j:=j+3;
        end;
      child1:=joop1;
      child2:=joop2;

end;
```

Pada prosedur diatas merupakan variasi dari *Position Based CrossOver*, perbedaannya terletak pada penempatan *substring* dari *parent* yang pertama yang ditempatkan sesuai order atau *substring* yang belum ada di *protochild* dari posisi yang telah ditempati *substring* dari *parent* kedua.

Fungsi Mutasi *Displacement* pada form *Genetika*

```
function Tgenetika.mutasi_displacement(induk:string):string;
var
i,j,pos_mutasi1,pos_mutasi2:integer;
nobar1,nobar2,child : string;
begin
j:=1;
pos_mutasi1:=random(jmlbrg);
repeat
  pos_mutasi1:=random(jmlbrg);
until pos_mutasi1 <> 0;
pos_mutasi2:=random(jmlbrg);
repeat
  pos_mutasi2:=random(jmlbrg);
  repeat
    pos_mutasi2:=random(jmlbrg);
  until pos_mutasi2 <> 0;
until pos_mutasi1 <> pos_mutasi2;
if pos_mutasi1 < 10 then
  nobar1 := '0' + inttostr(pos_mutasi1 )
else
  nobar1 := inttostr(pos_mutasi1);
if pos_mutasi2 < 10 then
  nobar2 := '0' + inttostr(pos_mutasi2 )
else
  nobar2 := inttostr(pos_mutasi2);
//laporan.RE.Lines.Add(nobar1 + ' , ' + nobar2);
```

```
while j <= length(induk) do
  begin
    if (induk[j] + induk[j+1]) = nobar1 then
         child := child + nobar2 + induk[j+2]
    else if (induk[j] + induk[j+1]) = nobar2 then
         child := child + nobar1 + induk[j+2]
    else
         child := child + induk[j] + induk[j+1] + induk[j+2];
    j := j+3;
 end;
mutasi_displacement:=child;
end;
```

Prosedur diatas digunakan untuk menjalankan metode Mutasi *Displacement*. Pada prosedur ini prinsipnya sama dengan metode mutasi *insertion*, perbedaannya terletak dari jumlah gen yang disisipkan, akan tetapi posisi penempatan tidak berubah.

Fungsi Mutasi *rotasi_posisi* pada form *Genetika*

```
function tgenetika.rotasi_posisi(induk : string):string;
var i,j,pos_rotasi : integer;
    brg_rotasi : string;
    child :  string;
begin
      pos_rotasi:=random(jmlbrg);
   repeat
     pos_rotasi:=random(jmlbrg);
   until pos_rotasi <> 0;
   j:=1;
   if pos_rotasi < 10 then
      brg_rotasi := '0' + inttostr(pos_rotasi)
   else
      brg_rotasi := inttostr(pos_rotasi);
       while j <= length(induk) do
       begin
          if (induk[j] + induk[j+1]) = brg_rotasi then
           begin
            if induk[j+2] = 'V' then
              child := child + induk[j] + induk[j+1] + 'H'
            else
              child := child + induk[j] + induk[j+1] + 'V';
           end
          else
            child := child + induk[j] + induk[j+1] + induk[j+2];
          j := j+3;
       end;
rotasi_posisi:=child;
end;
```

```
((ordinaty + panjang > ' + inttostr(y) + ' and pos = "H") or
(ordinaty + lebar > ' + inttostr(y) + ' and pos = "V"))))
and packed = "Y")';
qtempat.SQL.Clear;
qtempat.SQL.Add(strsql);
qtempat.active:=true;
//perhatikan posisi
if qtempat.FieldByName('Pos').AsString = 'H' then
ljoop := qtempat.FieldByName('lebar').AsInteger
else
ljoop := qtempat.FieldByName('panjang').AsInteger;
xjoop:= qtempat.FieldByName('ordinatx').AsInteger;
//jika tidak ada barang
if qtempat.recordcount = 0 then
begin
strsql:= 'select * from tbarang where ((ordinaty ='+
inttostr(y) +' and ordinatx>= '+ inttostr(x+ljoop) +' and
ordinatx < '+ inttostr(x+ljoop+l) + ') ';
strsql:= strsql +' or (ordinaty <'+ inttostr(y) +' and
((ordinaty + panjang > ' + inttostr(y) + ' and pos = "H") or
(ordinaty + lebar > ' + inttostr(y) + ' and pos = "V") ) and
ordinatx>= '+ inttostr(x+ljoop) +' and ordinatx < '+
inttostr(x+ljoop+l) + ')) and packed = "Y"';
qtempat.SQL.Clear;
qtempat.SQL.Add(strsql);
qtempat.active:=true;
//perhatikan posisi
if qtempat.FieldByName('pos').AsString = 'H' then
ljoop := qtempat.FieldByName('lebar').AsInteger
else
ljoop := qtempat.FieldByName('panjang').AsInteger;
xjoop:= qtempat.FieldByName('ordinatx').AsInteger;
//jika tidak ada
if qtempat.RecordCount = 0 then
begin
//ngecek diatasnya ada barang atau gak
strsql:= 'select * from tbarang where ((ordinatx =' +
inttostr(x) + ') or (ordinatx > ' + inttostr(x) + ' and
ordinatx < ' + inttostr(x+l) + ') or (ordinatx < ' +
inttostr(x) + ' and ((ordinatx + lebar > ' + inttostr(x) +
' and pos = "H") or (ordinatx + panjang > ' + inttostr(x) +
' and pos = "V" )) )) ';
strsql:= strsql + ' and ordinaty > ' + inttostr(y) + ' and
ordinaty <' + inttostr(y+p);
qtempat.SQL.Clear;
qtempat.SQL.Add(strsql);
qtempat.active:=true;
//perhatikan posisi
if qtempat.FieldByName('pos').AsString = 'H' then
ljoop := qtempat.FieldByName('lebar').AsInteger
else
ljoop := qtempat.FieldByName('panjang').AsInteger;
xjoop:= qtempat.FieldByName('ordinatx').AsInteger;
//jika tidak ada
if qtempat.RecordCount = 0 then
 begin
```

```
        qjoop.SQL.Clear;
        qjoop.SQL.Add('update tbarang set ordinatx = ' +
        inttostr(x) + ',ordinaty =' + intlostr(y) + ',packed =
        "Y" where kode_barang=' + quotedstr(kode));
        qjoop.ExecSQL;
        ketemu := true;
      end
        else
           x:= xjoop + ljoop;
              end
              else
              begin
                 x:= xjoop + ljoop;
              end;
           end
           else
           begin
              x:= xjoop + ljoop;
           end;
           end
           else
           begin
              x:= xjoop + ljoop;
           end;

        end;
        y:= y + 1;
      end;
      end;
```

Pada prosedur diatas digunakan untuk penempatan barang yang masuk kedalam gudang dengan memenuhi beberapa syarat yang harus dipenuhi agar barang dapat masuk dengan metode bottom-left.

5.6.2.2 Prosedure Pada Form Visual

fungsi dibawah merupakan fungsi untuk menggambar barang pada gudang

```
function CreateControl(ControlClass: TControlClass;
 constControlName:string;X,Y,W,H:Integer;induk:twinControl):
 TControl;
begin
 Result := ControlClass.Create(FVisual);
  with Result do
  begin
    Parent := induk;
    Name := ControlName;
    SetBounds(x,y ,w,h);

    Visible := True;
    sendtoback;
```

```
    result.
    repaint;
  end;
end;
```

Fungsi diatas untuk menggambar barang pada panel induk sesuai dengan variabel masukkan barang yang akan diproses.

Prosedure mengaktifkan visual barang

```
procedure TFVisual.FormActivate(Sender: TObject);
var
 i,j,x,y,w,h:integer;
 NAMAPANEL :string;
begin
  qvis.SQL.Clear;
  qvis.SQL.Add('select * from tgudang');
  qvis.Active:=True;
  panel1.Height :=qvis.fieldbyname('panjang').AsInteger * 15;
  panel1.width :=qvis.fieldbyname('lebar').AsInteger * 15;
  qvis.SQL.Clear;
  qvis.SQL.Add('select * from tbarang where packed = ' +
  quotedstr('Y'));
  qvis.Active:=True;
while not qvis.Eof do
 begin
  //Pemberian Kode barang
  namapanel := 'B' + qvis.fieldbyname('kode_barang').AsString ;
  x:=qvis.fieldbyname('ordinatx').AsInteger * 15 ;
  y:=qvis.fieldbyname('ordinaty').AsInteger * 15 ;
 //Penentuan Posisi barang pada gudang
 if qvis.fieldbyname('pos').Asstring  = 'V' then
  begin
   w:=qvis.fieldbyname('panjang').AsInteger  ;
   h:=qvis.fieldbyname('lebar').AsInteger ;
 end
 else if qvis.fieldbyname('pos').Asstring  = 'H' then
 begin
  h:=qvis.fieldbyname('panjang').AsInteger ;
  w:=qvis.fieldbyname('lebar').AsInteger ;
 end;
  w:= w * 15;
  h:=h * 15;
  //memanggil fungsi Createconrol untuk menggambar barang
 createcontrol(tpanel,namapanel,x,panel1.Height - (y +
 h),w,h,Panel1  );
 qvis.Next;
 end;
 end;
end.
```